



# ConFuse: Convolutional Transform Learning Fusion Framework For Multi-Channel Data Analysis

Pooja Gupta, Jyoti Maggu, Angshul Majumdar, Emilie Chouzenoux, Giovanni Chierchia

## ► To cite this version:

Pooja Gupta, Jyoti Maggu, Angshul Majumdar, Emilie Chouzenoux, Giovanni Chierchia. ConFuse: Convolutional Transform Learning Fusion Framework For Multi-Channel Data Analysis. EUSIPCO 2020 - 28th European Signal Processing Conference, Jan 2021, Amsterdam / Virtual, Netherlands. hal-02943658

**HAL Id: hal-02943658**

**<https://hal.science/hal-02943658>**

Submitted on 20 Sep 2020

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# ConFuse: Convolutional Transform Learning Fusion Framework For Multi-Channel Data Analysis

1<sup>st</sup> - Pooja Gupta

*Computer Science and Engineering*  
*Indraprastha Institute of Information Technology Delhi*  
Delhi, India  
poojg@iiitd.ac.in

2<sup>nd</sup> - Jyoti Maggu

*Computer Science and Engineering*  
*Indraprastha Institute of Information Technology Delhi*  
Delhi, India  
jyotim@iiitd.ac.in

3<sup>rd</sup> - Angshul Majumdar

*Electronics and Communications Engineering*  
*Indraprastha Institute of Information Technology Delhi*  
Delhi, India  
angshul@iiitd.ac.in

4<sup>th</sup> - Emilie Chouzenoux

*CVN, Inria Saclay*  
*Univ. Paris-Saclay*  
Gif-sur-Yvette, France  
emilie.chouzenoux@centralesupelec.fr

5<sup>th</sup> - Giovanni Chierchia

*LIGM, CNRS, ESIEE Paris*  
*Univ. Gustave Eiffel*  
Marne-la-Vallée, France  
giovanni.chierchia@esiee.fr

**Abstract**—This work addresses the problem of analyzing multi-channel time series data by proposing an unsupervised fusion framework based on convolutional transform learning. Each channel is processed by a separate 1D convolutional transform; the output of all the channels are fused by a fully connected layer of transform learning. The training procedure takes advantage of the proximal interpretation of activation functions. We apply the developed framework to multi-channel financial data for stock forecasting and trading. We compare our proposed formulation with benchmark deep time series analysis networks. The results show that our method yields considerably better results than those compared against.

**Index Terms**—CNN, transform learning, information fusion, stock forecasting and trading, finance data processing.

## I. INTRODUCTION

Several real-world problems can be expressed as a multi-channel time series analysis. Consider for instance the problem of demand forecasting [1], which consists of estimating the power consumption at a future point given the available information until the current instant. In building-level forecasting [2], typical inputs are power consumption, temperature, humidity, and occupancy, which can be considered as separate channels of a time series. Biomedical signal processing also involves multi-channel sequences, for instance in the problem of blood pressure (systolic and diastolic) estimation, the inputs are usually both the electrocardiogram (ECG) and the pulse plethysmogram (PPG) signals, forming three channels (2 for ECG and 1 for PPG) [3]. The last example, that we will particularly look at in this work, is stock price forecasting, where the next day close price must be predicted from five inputs of the current day.

Until a few years back, the standard deep learning approach for modeling time series data was based on long short-term memory (LSTM) [4] or gated recurrent unit (GRU)

[5]. Although theoretically appropriate, LSTM and GRU are hardware intensive and more time and memory consuming than 1D CNN. Moreover, LSTM and GRU could not model very long sequences. Owing to these shortcomings, 1D CNNs are gradually replacing LSTM, GRU and other RNN variants<sup>1</sup>. Another class of methods convert time series data to a matrix form and use 2D CNNs instead [6–8]. This 2D CNN model is especially popular in financial data analysis [9–11].

Deep learning has also been widely used for analyzing multi-channel / multi-sensor signals. In several such studies, all the sensors are stacked one after the other to form a matrix and 2D CNN is used for analyzing these signals [12, 13]. It is worth mentioning that most existing works address inference from multi-channel data using a supervised end-to-end machine learning paradigm [6–8, 10–14], where the input is the raw multi-channel signal and the output is the inferred parameter (e.g., class label, regression value). A recent alternative has been proposed in [15], relying on an unsupervised method that learns feature representations from multi-channel data, and then uses these features as an input of a suitable classifier or regression method, adapted to the user's need. Such an unsupervised strategy may offer greater flexibility than supervised ones. Consider for instance the problem of ECG data classification, involving either four or sixteen classes [16]. In the said work, two distinct deep neural networks are trained to solve both cases. In contrast, if the learning paradigm was unsupervised, only one network would be needed, providing as an output relevant features, which could, later on, serve as input to a third-party classifier or regressor.

In this work, we focus on two important problems of stock analysis, namely stock forecasting and stock trading. The former is a regression problem where the task is to predict the value of a stock, and the latter amounts to decide whether

This work was supported by the CNRS-CEFIPRA project under grant NextGenBP PRC2017.

<sup>1</sup><https://towardsdatascience.com/the-fall-of-rnn-lstm-2d1594c74ce0>

to buy or sell a stock. Instead of creating two separate end-to-end models, for performing classification and regression, we propose to learn a single unsupervised model. Motivated by the success of 1D CNNs for time series processing, coupled with the need for an unsupervised representation learning tool, we propose to make use of our recently introduced convolutional transform learning (CTL) approach [9]. Each channel is hence processed by 1D CTL and the resulting representations are fused by a fully connected network of transform learning, leading to the so-called *ConFuse* approach. An original end-to-end training strategy is proposed, that takes advantage of a key property of activation functions, thus allowing the use of the efficient stochastic procedure for learning the architecture parameters. Comparisons with state-of-the-art methods illustrate the benefits of the proposed method.

The rest of the paper will be structured as follows. Section 2 introduces the CTL representation learning machinery and presents the proposed method *ConFuse* for unsupervised multi-channel fusion. Experimental results are provided in Section 3. Conclusions are discussed in Section 4.

## II. PROPOSED FORMULATION

As already mentioned in the introduction, our approach consists in introducing a novel unsupervised framework for multi-channel data representation learning. A key ingredient of the latter will be the recently introduced CTL [9]. For the sake of clarity, we recall first the main steps of the CTL technique. Then, we will present the overall *ConFuse* architecture and discuss its training.

### A. Convolutional transform learning

CTL aims at learning a set of  $M$  convolutional filters  $(t_m)_{1 \leq m \leq M}$  from  $K$  observed samples  $(s^{(k)})_{1 \leq k \leq K}$ , so as to generate a set of  $MK$  feature vectors  $(x_m^{(k)})_{1 \leq k \leq K, 1 \leq m \leq M}$  representing adequately the data. The representation learning model is expressed as

$$\forall m \in \{1, \dots, M\}, \forall k \in \{1, \dots, K\}, t_m * s^{(k)} = x_m^{(k)}.$$

The training consists of learning  $(t_m)_{1 \leq m \leq M}$  and  $(x_m^{(k)})_{1 \leq k \leq K, 1 \leq m \leq M}$  from the dataset  $(s^{(k)})_{1 \leq k \leq K}$ . This is performed by solving the optimization problem defined as:

$$\begin{aligned} \underset{(t_m), (x_m^{(k)})}{\text{minimize}} \quad & \frac{1}{2} \sum_{k=1}^K \sum_{m=1}^M \left( \left\| t_m * s^{(k)} - x_m^{(k)} \right\|_2^2 + \psi(x_m^{(k)}) \right) \\ & + \mu \|T\|_F^2 - \lambda \log \det(T), \end{aligned} \quad (1)$$

where  $T = [t_1 | \dots | t_M]$  concatenates the filters,  $\det(T)$  is its determinant<sup>2</sup>, and  $\psi$  penalizes the feature vectors. Equivalently, using matricial notations, one should solve:

$$\begin{aligned} \underset{T, X}{\text{minimize}} \quad & \frac{1}{2} \sum_{k=1}^K \left( \|S_k T - X_k\|_F^2 + \Psi(X_k) \right) \\ & + \mu \|T\|_F^2 - \lambda \log \det(T). \end{aligned} \quad (2)$$

<sup>2</sup>Throughout the paper, we do not assume that the matrix  $T$  is squared. The determinant is computed from its singular value decomposition, following [17, Prop. 24.68]

We introduced the notations:

$$X_k = [x_1^{(k)} | \dots | x_M^{(k)}], \quad S_k T = [t_1 * s^{(k)} | \dots | t_M * s^{(k)}],$$

with  $S_k$  being the Toeplitz matrix associated to the 1D convolution product with filter  $s^{(k)}$  (assuming circulant padding). Moreover,  $\Psi$  applies the penalty term  $\psi$  column-wise on  $X_k$ , so that  $\Psi(X_k) = \sum_{m=1}^M \psi(x_m^{(k)})$ , and  $X = [X_1^\top | \dots | X_K^\top]^\top$ . The CTL strategy uses specific penalization terms over  $T$  and  $X$ . The penalty term on matrix  $T$  promotes the diversity among the learnt filters, something which is not guaranteed in CNN. Function  $\Psi$  can serve to impose some sparsity prior on  $X$  so as to avoid over-fitting [18, 19]. A local minimizer to Problem (2) can be obtained using the proximal alternating algorithm [20–22], which alternates between proximal updates on  $T$  and  $X$ . For more details on the updates derivations and the convergence guarantees, the readers can refer to [9].

### B. Proposed ConFuse Method

We are now ready to present our novel approach *ConFuse* for the unsupervised construction of representation features of multi-channel data. A natural strategy is to learn, for each channel  $c \in \{1, \dots, C\}$ , a distinct set of convolutional filters  $(T^{(c)})_{1 \leq c \leq C}$  and associated features  $(X^{(c)})_{1 \leq c \leq C}$ , by solving a CTL-based formulation:

$$\begin{aligned} \underset{T^{(c)}, X^{(c)}}{\text{minimize}} \quad & \frac{1}{2} \sum_{k=1}^K \left( \|S_k^{(c)} T^{(c)} - X_k^{(c)}\|_F^2 + \Psi(X_k^{(c)}) \right) \\ & + \mu \|T^{(c)}\|_F^2 - \lambda \log \det(T^{(c)}). \end{aligned} \quad (3)$$

Then, the learnt channel-wise features are stacked as  $X_k = [X_k^{(1)} | \dots | X_k^{(C)}]^\top$  for each  $k$ , and fused by a transform learning procedure acting as a fully-connected layer:

$$\begin{aligned} \underset{\tilde{T}, Z}{\text{minimize}} \quad & \frac{1}{2} \sum_{k=1}^K \|\tilde{T} X_k - Z_k\|_F^2 + \iota_+(Z) \\ & + \mu \|\tilde{T}\|_F^2 - \lambda \log \det(\tilde{T}), \end{aligned} \quad (4)$$

where  $\tilde{T}$  denotes the fusion stage transform (not assumed to be convolutional),  $Z$  is the row-wise concatenation of the fusion stage features  $(Z_k)_{1 \leq k \leq K}$ , and  $\iota_+$  is the indicator function for positive orthant, equals to zero if all the entries of  $Z$  are non-negative, and  $+\infty$  otherwise. Such non-negativity constraint allows to avoid trivial solutions.

However, the disjoint resolution of Problems (3) and (4) may lead to unstable solutions that are too sensitive to initialization. We, thus, propose an alternative strategy where we learn all the variables in an end-to-end fashion by solving a joint optimization problem. To this aim, we rely on the key property that the solution  $(\hat{X}^{(c)})_{1 \leq c \leq C}$  of the CTL problem assuming fixed filters  $(T^{(c)})_{1 \leq c \leq C}$  can be reformulated as the simple application of an element-wise activation function, that is, for every  $k \in \{1, \dots, K\}$ ,

$$\begin{aligned} \hat{X}_k(T) &= \left[ \hat{X}_k^{(c)}(T) \right]_{1 \leq c \leq C} \\ &= \left[ \Phi(S_k^{(c)} T^{(c)}) \right]_{1 \leq c \leq C}, \end{aligned} \quad (5)$$

with  $\Phi$  the proximity operator of  $\Psi$  [23]. For example, if  $\Psi$  is the indicator function of the positive orthant, then  $\Phi$  identifies with the famous rectified linear unit (ReLU) activation function. Many other examples are provided in [23]. Consequently, we propose to plug Equation (5) into Problem (4), leading to our final *ConFuse* formulation:

$$\begin{aligned} \underset{T, \tilde{T}, Z}{\text{minimize}} \quad & \frac{1}{2} \sum_{k=1}^K \|\tilde{T} \hat{X}_k(T) - Z_k\|_F^2 + \iota_+(Z) + \mu \|\tilde{T}\|_F^2 \\ & + \mu \|T\|_F^2 - \lambda \left( \log \det(\tilde{T}) + \sum_{c=1}^C \log \det(T^{(c)}) \right). \end{aligned} \quad (6)$$

Although Problem (6) is still nonconvex, there are two notable advantages with this new formulation. First, we remark that, as soon as the involved activation function is smooth, all terms of the cost function in (6) are differentiable, except the indicator function. We can thus employ the accelerated stochastic projected gradient descent, Adam, from [24]. The latter makes efficient use of automatic differentiation, stochastic approximations to efficiently deal with large-size datasets. Second, any (sub-)differentiable activation function  $\Phi$  can be plugged into our model (5), for instance SELU [25] or Leaky ReLU [26]. This flexibility will play a key role in the performance, as shown in the experimental section.

An example of structure of the learnt *ConFuse* architecture<sup>3</sup> is shown in Figure 1. Note that our approach is completely unsupervised. Specifically, we replaced supervision by explicitly learning the features  $Z$ , on which we impose the non-negativity constraint to avoid trivial solutions. Regarding the representation filters stacked in matrices  $(T, \tilde{T})$ , the log-det regularization imposes a full rank on those. Thus, it helps to enforce the diversity and to prevent the degenerate solution ( $T = 0, X = 0, \tilde{T} = 0, Z = 0$ ). The Frobenius regularization ensures that the matrices entries remain bounded.

### III. EXPERIMENTAL EVALUATION

We evaluate the proposed method, *ConFuse*, on the real-world problems of stock price prediction, and trading under stock forecasting. Both can be formulated as a multi-channel time series analysis, where the inputs are  $C = 5$  raw variables: opening price, closing price, (day's) low(est) price, (day's) high(est) price, and net asset value (NAV). As financial data are 1D signals, we will make use of 1D convolutional filters in our CTL representation, and the learnt features will then be fused to produce the final representation.

Experiments are carried out on the data from the National Stock Exchange (NSE) of India. The dataset contains information of 150 stocks between 2014 and 2018. Companies available are from various sectors, such as information technology (TCS, INFY), automobile (HEROMOTOCO, TATA-MOTORS), banking (HDFCBANK, ICICIBANK).

We have compared *ConFuse* with two state-of-the-art deep learning based time series models, namely TimeNet [27] based on LSTM, and ConvTimeNet [28] based on CNN. These

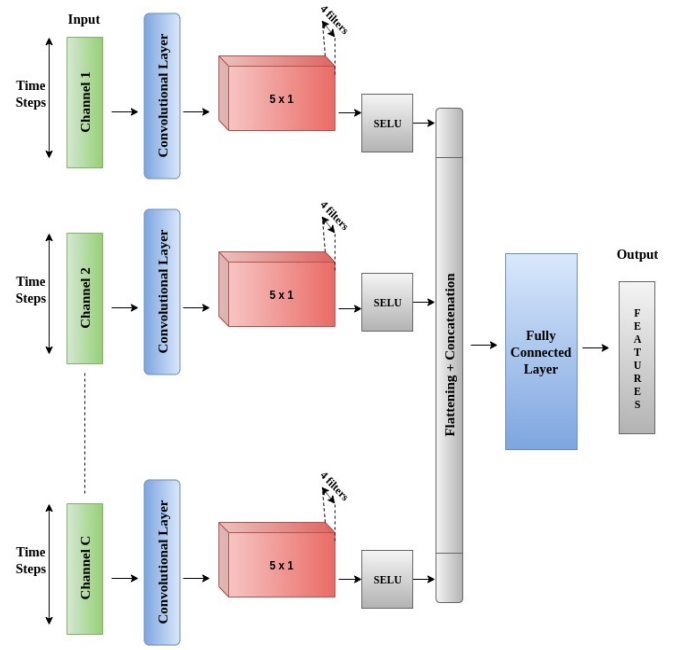


Fig. 1. Example of *ConFuse* architecture, for  $M = 4$  filters of size  $5 \times 1$ , and SELU activation function.

models were originally developed for univariate data. They were converted to multivariate form owing to our requirement of feeding five types of inputs. Each of the inputs is processed by either TimeNet or ConvTimeNet and the features from the five channels are fused by a fully connected layer with an output target. For the price forecasting problem, the target is multivariate – the outputs are the next day's variables (open, close, high, low and NAV). Since the outputs are real, the activation function at the output is linear. For the stock trading problem, the output is a binary BUY/SELL. Softmax is used to get the labels after the fully connected layer. The architectures of *ConFuse*, ConvTimeNet, and TimeNet are summarized in Table I. We have tested six different activation functions for *ConFuse*.<sup>4</sup>

The optimizer used for each architecture is Adam with parameters  $(\beta_1, \beta_2) = (0.9, 0.999)$  and  $\epsilon = 10^{-8}$ . Weight decay is set to  $5 \times 10^{-5}$  for *ConFuse* and TimeNet, and  $10^{-4}$  for ConvTimeNet. The regularization parameters  $\mu = 10^{-4}$  and  $\lambda = 10^{-2}$  are used in *ConFuse*, for all activation functions. These configurations give the best results.

The proposed technique, *ConFuse*, generates unsupervised features. In the forecasting problem, we input these features to ridge regression for forecasting. Evaluation is carried out in terms of mean absolute error (MAE) between the predicted and actual stock prices. The averages of MAE of all the stocks forecasting results are shown in Table II. For the stock trading problem, we apply a random decision forest to the generated

<sup>4</sup>The gradients of SELU, RELU, PRELU, and Leaky RELU are not defined in zero. It is customary to consider any valid sub-gradient value instead. We did not notice any practical convergence issues with ADAM algorithm when resorting to this strategy.

<sup>3</sup>Code available at - <https://github.com/pooja290992/ConFuse>

features for classification. We evaluate the results in terms of area under the ROC curve (AUC), precision, recall, F1 score and relative difference (between actual and predicted) in annualized return (AR).

TABLE I  
DESCRIPTION OF DIFFERENT TECHNIQUES

| Method      | Architecture Description  |
|-------------|---|
| ConFuse     | $5 \times \left\{ \begin{array}{l} \text{layer1: 1D Conv}(1, 4, 5, 1, 2)^{\dagger} \\ \text{Activation (e.g., ReLU)} \end{array} \right.$<br>$1 \times \text{layer2: Transform Learning}$   |
| ConvTimeNet | $5 \times \left\{ \begin{array}{l} \text{layer1: 1D Conv}(1, 32, 9, 1, 4)^{\dagger} \\ \text{Batch Norm + SELU} \\ \text{layer2: 1D Conv}(32, 32, 3, 1, 1)^{\dagger} \\ \text{Batch Norm + SELU + Skip conn.} \\ \text{layer3: 1D Conv}(32, 64, 9, 1, 4)^{\dagger} \\ \text{Batch Norm + SELU} \\ \text{layer4: 1D Conv}(64, 64, 3, 1, 1)^{\dagger} \\ \text{Batch Norm + SELU + Skip conn.} \\ \text{layer3: Global Average Pooling} \end{array} \right.$<br>$\text{layer4: Fully Connected}$<br>$\text{For Trading, added layer5: Softmax}$ |
| TimeNet     | $5 \times \left\{ \begin{array}{l} \text{layer1: LSTM unit} \\ (1, 12, 2, \text{True})^{\S} \\ \text{layer2: Global Average Pooling} \end{array} \right.$<br>$\text{layer3: Fully Connected}$<br>$\text{layer4: Softmax}$   |

<sup>†</sup> input bands, output bands, kernel size, stride, padding

<sup>§</sup> input size, hidden size, num. layers, bidirectional

TABLE II  
FORECASTING RESULTS (MAE)

| Method            | Open         | Close        | High         | Low          | NAV          |
|-------------------|--------------|--------------|--------------|--------------|--------------|
| ConFuse-SELU      | 0.011        | 0.023        | 0.017        | 0.017        | 0.447        |
| ConFuse-ReLU      | 0.009        | 0.021        | 0.014        | 0.014        | 0.445        |
| ConFuse-PReLU     | 0.007        | 0.017        | 0.012        | 0.013        | 0.434        |
| ConFuse-LeakyReLU | <b>0.007</b> | <b>0.017</b> | <b>0.012</b> | <b>0.013</b> | <b>0.427</b> |
| ConFuse-Tanh      | 0.258        | 0.259        | 0.258        | 0.259        | 0.488        |
| ConFuse-Sigmoid   | 0.227        | 0.227        | 0.227        | 0.227        | 0.482        |
| ConvTimeNet       | 1.551        | 1.554        | 1.535        | 1.567        | 2.357        |
| TimeNet           | 0.295        | 0.295        | 0.294        | 0.296        | 0.511        |

TABLE III  
TRADING RESULTS

| Method            | Precis.      | Recall       | F1           | AUC          | AR            |
|-------------------|--------------|--------------|--------------|--------------|---------------|
| ConFuse-SELU      | <b>0.524</b> | <b>0.777</b> | <b>0.619</b> | <b>0.543</b> | <b>17.898</b> |
| ConFuse-RELU      | 0.505        | 0.648        | 0.556        | 0.523        | 18.112        |
| ConFuse-PReLU     | 0.491        | 0.601        | 0.528        | 0.506        | 19.091        |
| ConFuse-LeakyReLU | 0.496        | 0.602        | 0.531        | 0.511        | 19.150        |
| ConFuse-Tanh      | 0.469        | 0.560        | 0.493        | 0.497        | 19.002        |
| ConFuse-Sigmoid   | 0.487        | 0.584        | 0.513        | 0.498        | 20.540        |
| ConvTimeNet       | 0.457        | 0.507        | 0.413        | 0.524        | 19.410        |
| TimeNet           | 0.469        | 0.648        | 0.496        | 0.513        | 18.764        |

We find that the results for the stock forecasting problem are exceptionally good. For most tested activation functions,

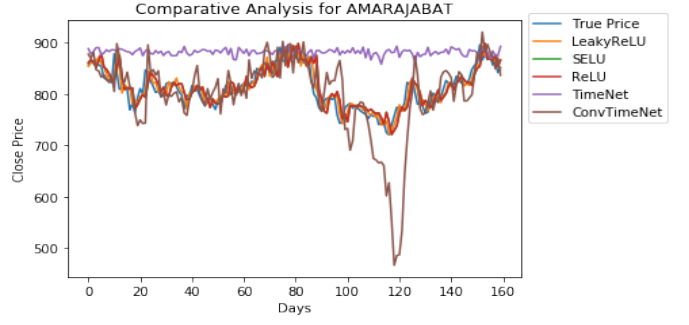


Fig. 2. Stock Forecasting Performance

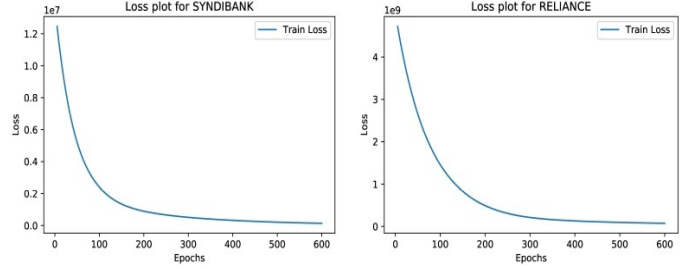


Fig. 3. Empirical Convergence Plots

*ConFuse* has MAE more than one order of magnitude lower than the state-of-the-arts. For the stock trading problem, *ConFuse* outperforms the benchmarks as well with two activation functions, namely SELU and ReLU, and reaches similar performance than the benchmarks with the other activation functions. Given the limitations in space, we can only show the average values over the 150 stocks. For the qualitative evaluation of a few stocks, we display example of forecasting result on a randomly chosen stock in Figure 2, which corroborates the very good performance of the proposed solution. We also show some empirical convergence plots of Adam in Figure 3 when using *ConFuse* with SELU, which depict the practical stability of our end-to-end training method.

## IV. CONCLUSION

This work proposes *ConFuse*, a novel multi-channel unsupervised fusion framework based on learnt convolutional filters. It overcomes the fundamental issue of convolutional nets, that is their inability to learn in an unsupervised fashion. Our work is based on the premise of CTL [9], a technique for learning convolutional filters in an unsupervised fashion. In the proposed fusion framework, each of the channels is processed by CTL. The outputs from these filters are concatenated and fused by fully-connected transform learning. An original end-to-end training strategy is employed. We find that the proposed technique improves over state-of-the-art deep learning based time series analysis techniques. Currently, our model only uses one layer of learnt convolutional filters. In future work, we would like to extend this work to a multi-layer architecture.

# REFERENCES

- [1] L. Suganthi and A. Samuel, "Energy models for demand forecasting—a review," *Renewable and Sustainable Energy Reviews*, vol. 16, no. 2, pp. 1223–1240, 2012.
- [2] B. Yildiz, J. Bilbao, and A. Sproul, "A review and analysis of regression and machine learning models on commercial building electricity load forecasting," *Renewable and Sustainable Energy Reviews*, vol. 73, pp. 1104–1122, 2017.
- [3] Y. Yoon, J. Cho, and G. Yoon, "Non-constrained blood pressure monitoring using ECG and PPG for personal healthcare," *Journal of Medical Systems*, vol. 33, no. 4, pp. 261–266, 2009.
- [4] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [5] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," in *Proc. of NIPS Workshop on Deep Learning*, Montreal, Canada, 8-13 Dec. 2014.
- [6] N. Hatami, Y. Gavet, and J. Debayle, "Classification of time-series images using deep convolutional neural networks," in *Proc. of ICMV*, Vienna, Austria, 13-15 Nov. 2017.
- [7] Z. Wang and T. Oates, "Imaging time-series to improve classification and imputation," in *Proc. of IJCAI*, Buenos Aires, Argentina, 25-31 July 2015.
- [8] O. Sezer and A. Ozbayoglu, "Algorithmic financial trading with deep convolutional neural networks: Time series to image conversion approach," *Applied Soft Computing*, vol. 70, pp. 525–538, 2018.
- [9] J. Maggu, E. Chouzenoux, G. Chierchia, and A. Majumdar, "Convolutional transform learning," in *Proc. of ICONIP*, Siem Reap, Cambodia, 13-16 Dec. 2018, pp. 162–174.
- [10] A. Tsantekidis, N. Passalis, A. Tefas, J. Kannianen, M. Gabbouj, and A. Iosifidis, "Forecasting stock prices from the limit order book using convolutional neural networks," in *Proc. of CBI*, Thessaloniki, Greece, 24-27 July 2017.
- [11] M. Gudelek, S. Boluk, and A. Ozbayoglu, "A deep learning based stock trading model with 2D CNN trend detection," in *Proc. of SSCL*, Hawaii, USA, 27 Nov. - 1 Dec. 2017.
- [12] J. Yang, M. Nguyen, P. San, X. Li, and S. Krishnaswamy, "Deep convolutional neural networks on multichannel time series for human activity recognition," in *Proc. of IJCAI*, Buenos Aires, Argentina, 25-31 July 2015.
- [13] S. Yao, S. Hu, Y. Zhao, A. Zhang, and T. Abdelzaher, "A unified deep learning framework for time-series mobile sensing data processing," in *Proc. of WWW*, Montreal, Canada, 11-15 Apr. 2016, pp. 351–360.
- [14] Y. Zheng, Q. Liu, E. Chen, Y. Ge, and J. Zhao, "Time series classification using multi-channels deep convolutional neural networks," in *Proc. of WAIM*, Macau, China, 16-18 June 2014.
- [15] S. Löwe, P. O'Connor, and B. Veeling, "Putting an end to end-to-end: Gradient-isolated learning of representations," in *Proc. of NeurIPS*, Vancouver, Canada, 8-14 Dec. 2019, pp. 3033–3045.
- [16] A. Hannun, P. Rajpurkar, M. Haghpanahi, G. Tison, C. Bourn, M. Turakhia, and A. Ng, "Cardiologist-level arrhythmia detection and classification in ambulatory electrocardiograms using a deep neural network," *Nature Medicine*, vol. 25, no. 1, p. 65, 2019.
- [17] H. H. Bauschke and P. L. Combettes, *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*. Springer, New York, 2017, 2nd edition.
- [18] S. Ravishanker and Y. Bresler, "Learning sparsifying transforms," *IEEE Transactions on Signal Processing*, vol. 61, no. 5, pp. 1072–1086, 2012.
- [19] L. M. Briceno-Arias, G. Chierchia, E. Chouzenoux, and J.-C. Pesquet, "A random block-coordinate douglas-rachford splitting method with low computational complexity for binary logistic regression," *Computational Optimization and Applications*, vol. 72, no. 3, pp. 707–726, Jan. 2019.
- [20] E. Chouzenoux, J.-C. Pesquet, and A. Repetti, "A block coordinate variable metric forward-backward algorithm," *Journal of Global Optimization*, vol. 66, no. 3, pp. 457–485, 2016.
- [21] J. Bolte, S. Sabach, and M. Teboulle, "Proximal alternating linearized minimization for nonconvex and nonsmooth problems," *Mathematical Programming*, vol. 146, no. 1–2, pp. 459–494, 2014.
- [22] H. Attouch, J. Bolte, and B. F. Svaiter, "Convergence of descent methods for semi-algebraic and tame problems: proximal algorithms, forward-backward splitting, and regularized Gauss-Seidel methods," *Mathematical Programming*, vol. 137, no. 1–2, pp. 91–129, 2013.
- [23] P. Combettes and J.-C. Pesquet, "Deep neural network structures solving variational inequalities," *Set-Valued and Variational Analysis*, 2020, <https://arxiv.org/abs/1808.07526>.
- [24] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. of ICLR*, 2015.
- [25] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-normalizing neural networks," in *Proc. of NeurIPS*, Long Beach, California, USA, 4-9 Dec. 2017.
- [26] A. Mass, A. Hannun, and A. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. of ICML*, Atlanta, USA, 16-21 June 2013.
- [27] P. Malhotra, V. TV, L. Vig, P. Agarwal, and G. Shroff, "Timenet: Pre-trained deep recurrent neural network for time series classification," in *Proc. of ESANN*, Bruges, Belgium, 26-28 Apr. 2017.
- [28] K. Kashiparekh, J. Narwariya, P. Malhotra, L. Vig, and G. Shroff, "ConvTimeNet: A pre-trained deep convolutional neural network for time series classification," *Tech. Rep.*, 2019, <https://arxiv.org/abs/1904.12546>.